# Don't Let Discourse Confine Your Model: Sequence Perturbations for Improved Event Language Models

**Mahnaz Koupaee**[1], **Greg Durrett**[2], **Nathanael Chambers**[3], **Niranjan Balasubramanian**[1]

[1] Stony Brook University, [2] The University of Texas at Austin, [3] United States Naval Academy
[1]{mkoupaee,niranjan}@cs.stonybrook.edu
[2]gdurrett@cs.utexas.edu, [3]nchamber@usna.edu

## Abstract

Event language models represent plausible sequences of events. Most existing approaches train autoregressive models on text, which successfully capture event co-occurrence but unfortunately constrain the model to follow the *discourse order* in which events are presented. Other domains may employ different discourse orders, and for many applications, we may care about different notions of ordering (e.g., temporal) or not care about ordering at all (e.g., when predicting related events in a schema). We propose a simple yet surprisingly effective strategy for improving event language models by perturbing event sequences so we can relax model dependence on text order. Despite generating completely synthetic event orderings, we show that this technique improves the performance of the event language models on both applications and out-of-domain events data.

## 1 Introduction

Event-level language models (LMs) provide a way to reason about events, and to approximate schematic and script-like knowledge (Schank and Abelson, 1977; Balasubramanian et al., 2013; Nguyen et al., 2015) about them (Modi and Titov, 2014; Pichotta and Mooney, 2016; Weber et al., 2018). These models aim to learn high-level representations of complex events (e.g., an arrest) and possibly their entity roles from raw text (e.g., a suspect). However, a major limitation is their reliance on the *discourse* order of event mentions when training the LM. Although powerful, these event LMs capture information we don't want in true world knowledge. For instance, a script of events may be weakly ordered in real life, but the system instead learns to strongly rely on the text order in which the events were described. Figure 1 shows an example where discourse and actual temporal order are different: a model trained on newswire may learn the pattern on the left from obituaries, but will fail to generalize to biographical or other narrative descriptions of someone's life.

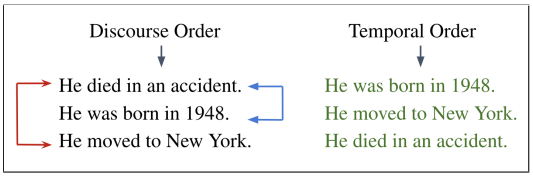


Figure 1: Example of an event schema for which the discourse order is different from the temporal order.

In this paper, we aim to improve event-level LMs in order to make them more suitable for general knowledge learning. While a range of possible modifications to the model can be imagined, such as set transformers (Lee et al., 2019), we want to leverage autoregressive pre-trained LMs. We instead find that we can encode the necessary invariances via data augmentation: namely, we apply a set of event sequence perturbations to sequences in the training data to relax the model's dependence on discourse order. By considering the next event based on shuffled sequences of events, we encourage the model to treat the input more as a set of events rather than strictly as a discourse sequence.

Surprisingly, despite our disruption of discourse order, experiments show how perturbations can improve event language modeling of text, particularly when evaluating the model on other domains which present events in different orders (e.g., novels or blogs present data in more of a "narrative" fashion than news datasets common in NLP (Yao and Huang, 2018)). Our experiments evaluate accuracy on the Inverse Narrative Cloze task on in-domain newswire, as well as out-domain novels and blogs[1].

[1]The code and data is available at https://github.com/StonyBrookNLP/elm-perturbations



*Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Short Papers)*, pages 599–604
August 1–6, 2021. 

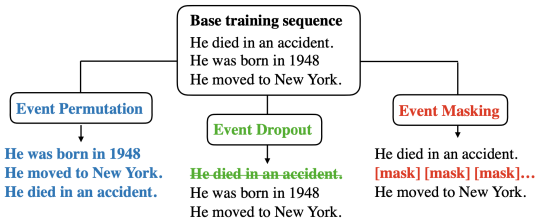


Figure 2: Sequence perturbations strategies.

## 2 Perturbing Discourse Sequences

Event language modeling tasks are typically defined over sequences of events as they appear in text. The events can be represented either as a sequence of words annotated with predicate-argument structure (e.g., semantic roles (Pichotta and Mooney, 2016), Open IE tuples (Weber et al., 2018; Rudinger et al., 2015) or with compositional embeddings (Modi, 2016). Generative models are trained to predict subsequent events in a sequence conditioning on previously observed events. Naturally, these models learn the order in which events appeared in text (Manshadi et al., 2008).

However, relying on discourse order may not be necessary and can potentially limit generalization of event LMs. For some event related tasks such as schema learning (Weber et al., 2018), the discourse order is not directly relevant. For other tasks such as event ordering (Pustejovsky et al., 2003; Chambers et al., 2014; Wang et al., 2018), temporal or logical order of events is most critical – discourse order, at best, is a noisy proxy. In fact, the first systems for schema learning were noticeably *not* language models (Mooney and DeJong, 1985; Chambers and Jurafsky, 2009, 2011). We introduce three simple perturbation techniques shown in Figure 2 that relax the reliance on discourse sequences.

### 2.1 Event Permutation

One way to reduce reliance on discourse order is to expose the model to random permutations of the input sequences, as shown in Figure 2. Using all possible permutations of a sequence is impractical, so we introduce three specific shuffles that force the model to pay attention to long-term dependencies and avoid the over-reliance on local dependencies/order:

- Reversed order: given a set of events as `ABCD`, the reverse of the sequence is created as `DCBA`.
- Concatenation of events in the odd positions followed by the even positions of the sequence: the permuted sequence is `BDAC`.
- Concatenation of event tuples in the odd positions followed by those in the even positions of the *reverse* order of the original sequence. The new sequence is: `CADB`

These shuffle patterns were selected to minimize the chance of repetition across permutations.

### 2.2 Event Dropout

We also consider event dropout as another perturbation to the original discourse sequence. For each sequence, we remove a small random subset of events (Event Dropout in Figure 2). We create multiple reduced sequences for each original sequence. The reduced sequences are treated in the same way as the original sequences for training the model. This perturbation is a type of regularization against overfitting on any specific event in a sequence, much like standard dropout procedures.

### 2.3 Event Masking

When dropping events, we can provide additional information to the model about where events were dropped. This forces the model to capture longer-term dependencies among events in the sequence. We randomly select a number of event tuples and replace their tokens with a `<mask>` token (Masking in Figure 2). For each sequence in the training set, we generate its masked sequences with each having a fixed proportion of its events masked.

## 3 Experimental Setup

**Data** We train event language models on the Annotated NYT corpus using Open IE event tuples extracted by Ollie (Schmitz et al., 2012). The dataset contains a total of around 1.8 million articles. After preprocessing steps, 1,467,366 articles are used as the training set, 6k articles as test set and 4k articles as the dev set. Each event is a 4-tuple `(v,s,o,p)` containing the verb, subject, object and preposition. We follow the same preprocessing steps outlined in Weber et al. (2018) to create event sequences.

The components of the events (the verb, subject, etc.) are all individual tokens, and are treated like normal text. For example, the events (truck packed with explosives), (police arrested suspect), would be given to the model as: packed truck explosives with [TUP] arrested police suspect _NULL_, where

_NULL_ is the null preposition token and [TUP] is a special separator token between events.

Each document is first partitioned into segments of four sentences each. All events extracted from each segment are concatenated (in discourse order) to form an event sequence. This is a simple heuristic to avoid considering event sequences that can drift or connect otherwise unrelated events. Tuples with common verbs (is, are, be, ...) and repeating predicates are also ignored.

The training, development, and test splits have 7.1M, 19K, and 29K event sequences respectively. During training, depending on the perturbation strategy used, a number of sequences are added to the initial sets. The numbers are hyperparameters, selected differently for each model. Details are given in the following sections.

**Autoregressive Models** Our baseline autoregressive event LM is a pretrained GPT-2 model (Radford et al., 2019) fine-tuned on the event sequences.

Once the perturbations are applied to the original sequence, the modified sequence is used as both the input and the output of the model. We trained variants of GPT-2 with different sequence perturbations as shown in Figure 2 in our experiments. For the dropout and masked versions, we created $n/3$ new sequences with $n$ being the number of events in the sequence. Each sequence has $n/3$ of its events either dropped or masked.

**Autoencoding Models** We use HierarchicAl Quantized Autoencoder (HAQAE) (Weber et al., 2018) as a strong autoencoding model. HAQAE is an LSTM-based autoencoder, which uses a hierarchical latent space to model event sequences. HAQAE uses categorical global latent variables to represent a tree-structured hierarchy which allow it to model different types of schemas and their possible tracks. Different levels of this hierarchical structure capture different levels of features of the schemas.

For training the HAQAE model, instead of reconstructing a perturbed sequence, we explore a denoising style training objective, where we *only* perturb the input part of the sequence keeping the output the same as the original. Our hypothesis is that these models learn a perturbation-invariant latent space representation in both cases, which will help break the dependence on discourse order. We use the denoising variant in our experiments as it worked better than the standard reconstruction objective in our initial experiments.

| | Type of System | PPL Val | PPL Test | INC Val | INC Test |
|---|---|---|---|---|---|
| | Random Baseline | - | - | 16.60 | 16.60 |
| Auto regressive | RNNLM | 91.84 | 90.92 | 25.30 | 26.30 |
| | GPT-2 Baseline | 85.13 | 84.13 | 26.80 | 28.30 |
| | GPT-2 Masked | 87.96 | 87.26 | 26.30 | 27.10 |
| | GPT-2 Dropout | 83.46 | 82.56 | 26.70 | 27.70 |
| | GPT-2 Permuted | **83.18** | **82.26** | **27.45** | **28.90** |
| Auto encoding | HAQAE-Baseline | 142.22 | 140.89 | 31.80 | 33.85 |
| | HAQAE-Masked | 148.07 | 147.03 | 33.80 | 36.80 |
| | HAQAE-Dropout | **122.69** | **122.30** | 31.25 | 32.25 |
| | HAQAE-Permuted | 143.39 | 142.07 | **34.75** | **38.55** |

Table 1: Perplexity and the accuracy of Inverse Narrative Cloze task. Lower is better for perplexity while higher is better for INC.

For each sequence in the permutation model, we generated permuted sequences for 10% of the original sequences. As for the dropout and masked models, we created $n/4$ new sequences with $n$ being the number of events in the sequence. Each sequence has $n/3$ of its events either dropped or masked. Preliminary experiments showed little difference between using all the data vs a subset.

**Models Hyperparameters** The GPT-2 model uses the implementation from Huggingface library (Wolf et al., 2020) using a pre-trained gpt-2 small model and tokenizer. Adam optimizer (Kingma and Ba, 2014) is used with an initial learning rate of $6.25e-5$.

The HAQAE model uses 5 discrete latent variables. Each variable can initially take on K = 512 values, with an embeddings dimension of 256. The encoder is a bidirectional, single layer RNN with GRU cell (Cho et al., 2014) with a hidden dimension of size 512. The embeddings size is 300 which are initialized with pretrained GloVe (Pennington et al., 2014) vectors. The decoder is also a single layer RNN with GRU cells with a hidden dimension of 512 and 300 dimensional word embeddings (initialized) as inputs. All experiments use a vocabulary size of 50k. Adam optimizer with a learning rate of 0.0005 is used.

## 4 Evaluation

We ran different experiments to answer the following questions:

**How do sequence perturbation techniques improve event language modeling?** We evaluate perplexity as is standard in Table 1, but aside from

| System | Blogs | Novels | News |
|---|---|---|---|
| HAQAE-baseline | 24.31 | 25.10 | 32.25 |
| **HAQAE-permuted** | **31.95** | **28.45** | **38.75** |

Table 2: INC accuracy on external data

| Legitimate Sequence | Confounding Sequence |
|---|---|
| ↓ | ↓ |
| he issued estimates | he issued estimates |
| he received extension | homes assess ability |
| candidate pledged before primary | department specify information |
| return release in august | provisions govern training |
| griffin president of investments | promise of laws unfulfilled |
| lauder pay income | promise remains unfulfilled |

Figure 3: A legitimate sequence and its confounding.

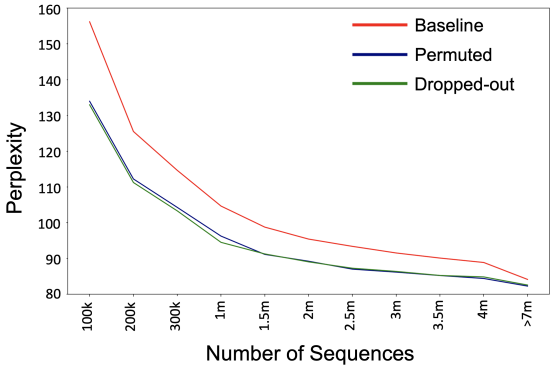


Figure 4: Perplexity of different GPT-2 models with respect to the number of training sequences.

perplexity, we want to see how well event LMs capture schematic knowledge. We thus evaluate on the inverse narrative cloze (INC) task (Weber et al., 2018). Given the first event from an original discourse sequence and a set of candidate event sequences, the task is to identify the true event sequence completion. This evaluation is closer to our ultimate goal: identifying realistic event schemas rather than discourse-focused metrics like perplexity.

The INC evaluation starts with a gold sequence of events from a real document, and then includes 5 other event sequences pulled from confounding documents. You insert the first gold event artificially at the start of each of these. The gold event sequence should have high probability compared to the confounding event sequences. Figure 3 shows a gold sequence and one confounding sequence generated for it. The six sequences are ranked based on the probabilities assigned by the model, and then the accuracy is the number of predictions where the gold sequence is ranked first. A random model will uniformly choose one among the six sequences and thus will have an accuracy of 16.6%.

The perplexity[2] and the INC accuracy of different variants of both autoregressive and autoencoding models are shown in Table 1.

Using sequence perturbations improves the INC accuracy on both test and validation sets for both categories of models. Further, the sequence perturbations gain in terms of INC accuracy is much higher with HAQAE.

[2]For autoencoders we report generative perplexity with the KL-term, while the original paper (Weber et al., 2018) has the lower reconstructive perplexity without the KL-term.

**How do models trained with perturbation techniques perform on out-of-domain data?** The NYT corpus used for training the models in this study is newswire. The journalistic writing style does not always follow the temporal ordering of events, but represents the events in various orders going backwards or forward in time. One might argue that the reason the sequence perturbations work better in terms of INC accuracy is that the events extracted from news do not necessarily follow the temporal order and therefore the perturbations will not create an issue. To show the effectiveness of our approach, we evaluated the performance of our models on the event sequences extracted from narratives coming from different domains: novels, blogs and news (Yao and Huang, 2018).

We used the OpenIE extraction system in a similar fashion to extract the event tuples from the narrative sequences. We used our best-performing model from the previous section and with no fine-tuning applied the models to see how our sequence perturbations performed in terms of INC accuracy on these narrative texts. The results of this analysis are presented in Table 2. The numbers show that the proposed sequence perturbations perform better on out-of-domain data (with explicit temporal links) compared to the baseline model.

**How effective are the sequence perturbation techniques with respect to the number of training instances?** Our sequence perturbations can be seen as data augmentation strategies which will help models learn new aspects of data that can not be learned from the original sequences. As the number of training samples increases, the model has more opportunities to learn these aspects. Therefore, the sequence perturbations will be more useful for domains with fewer training samples.

| | seed | generated events | ppl(g—s) | ppl(g—ps) |
|---|---|---|---|---|
| HAQAE-Baseline | people reported fire, fire spread to forest | people died in fire, fire caused fires | 4.49 | 6.49 |
| | fire spread to forest, people reported fire | person spokesman for department, firefighters taken to hospital | 5.21 | 6.30 |
| HAQAE-permuted | people reported fire, fire spread to forest | fires began today, people working in area | 5.75 | 5.58 |
| | fire spread to forest, people reported fire | fires began today, people working in area | 5.58 | 5.75 |

Table 3: Generated schemas for two-event seeds. The second row for each model shows the generated schemas for permuted seed events. ppl(g—s) and ppl(g—ps) are the perplexity of generated events given the seed events and the perplexity of events given permuted seeds. The lower the difference the more robust the model is to permutations.

| | seed | generated events |
|---|---|---|
| HAQAE-baseline | fire spread to neighborhood, people reported fire | people fire to floor, person spokesman for department |
| | fire spread to forest, people reported fire | firefighters fire to floor, person spokesman for department |
| HAQAE-permuted | fire spread to neighborhood, people reported fire | Fire spread through floors, fire came from floor |
| | fire spread to forest, people reported fire | fires began today, people working in area |

Table 4: Generated schemas for two-event seeds. The second event is the same while the first event shows a different branch.

We plotted the perplexity with respect to the number of training sequences for the GPT-2 baseline system as well as permuted and dropout models. As can be seen in Figure 4, the gap between the perplexity scores are higher when the number of sequences are lower. This observation suggests that our approach will result in better language models for domains with limited data.

**How do schemas generated by different models differ from each other?** We generated schemas for 46 two-event seeds using the HAQAE baseline and permuted models. We wanted to see how the generated schemas differ in two different aspects: First, for each seed, we permuted the events and generated schemas for both models. We expect the permuted model to have less variation in generating events for original and permuted seeds. We calculated the perplexity of the generated events for both the original order of events as well as the permuted order. Table 3 shows an example of such scenario where the HAQAE-permuted model has lower variation in perplexity for permuted seed events.

Second, we want to see how dependent the generation is upon the most recent event in the sequence. We generated schemas for two-event seeds in which the last event is the same while the first event indicates a different path. Table 4 shows an example where the permuted model generates more diverse events.

## 5 Conclusion

We proposed a set of simple sequence perturbations to relax the model's reliance on the discourse order of event mentions for event language modeling. By predicting the next event based on perturbed sequences, the model is encouraged to treat the input as a *set* of events. Our experiments show that these perturbations can improve identifying event schemas measured by INC accuracy both on in-domain and out-of-domain data.

## Acknowledgments

We would like to thank Noah Weber for providing helpful directions on the HAQAE model. This material is based on research that is supported by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes.

## References

Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni, et al. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1721–1731.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense Event Ordering with a Multi-Pass Architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 976–986.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR.

Mehdi Manshadi, Reid Swanson, and Andrew S Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *FLAIRS Conference*, pages 159–164.

Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 75–83.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 49–57.

Raymond J Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *IJCAI*, pages 681–687.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 188–197.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Karl Pichotta and Raymond J Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*.

James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Rob Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003. The TimeBank corpus. *Proceedings of Corpus Linguistics*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686.

Roger C Schank and Robert P Abelson. 1977. Scripts, plans, goals, and understanding: an inquiry into human knowledge structures.

Michael Schmitz, Stephen Soderland, Robert Bart, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534.

Su Wang, Eric Holgate, Greg Durrett, and Katrin Erk. 2018. Picking apart story salads. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1455–1465, Brussels, Belgium. Association for Computational Linguistics.

Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Hierarchical quantized representations for script generation. *arXiv preprint arXiv:1808.09542*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wenlin Yao and Ruihong Huang. 2018. Temporal event knowledge acquisition via identifying narratives. *arXiv preprint arXiv:1805.10956*.